Optimization

Jungdam Won

Computer Science & Engineering

Seoul National Univ.

Agenda

- What is optimization?
  - Terminologies
- Methods to solve the optimization problem
  - Analytical methods
  - Numerical methods
- Some examples and special types of problems

What is Optimization?

- The selection of a best element, with regard to some criterion, from some set of available alternatives. -Wikipedia-

- Examples
  - How to design a meal with enough nutrition given limited budget?
  - What is the best VLSI (IC chip) design to minimize its size while satisfying the performance criteria?
  - What are the parameters of deep neural networks that provide the best classification performance?

What is Optimization?

$$\min_{x_1, x_2, \cdots, x_n} f(x_1, x_2, \cdots, x_n)$$

Variables
(a.k.a. parameters)

Loss function
(a.k.a. cost, energy, objective)

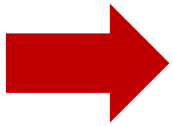
Find **a minimum value** of $f(x_1, x_2, \cdots, x_n)$
among all possible combination of variables $x_1, x_2, \cdots, x_n$

What is Optimization?

$$\underset{x_1, x_2, \cdots, x_n}{\operatorname{argmin}} \; f(x_1, x_2, \cdots, x_n)$$

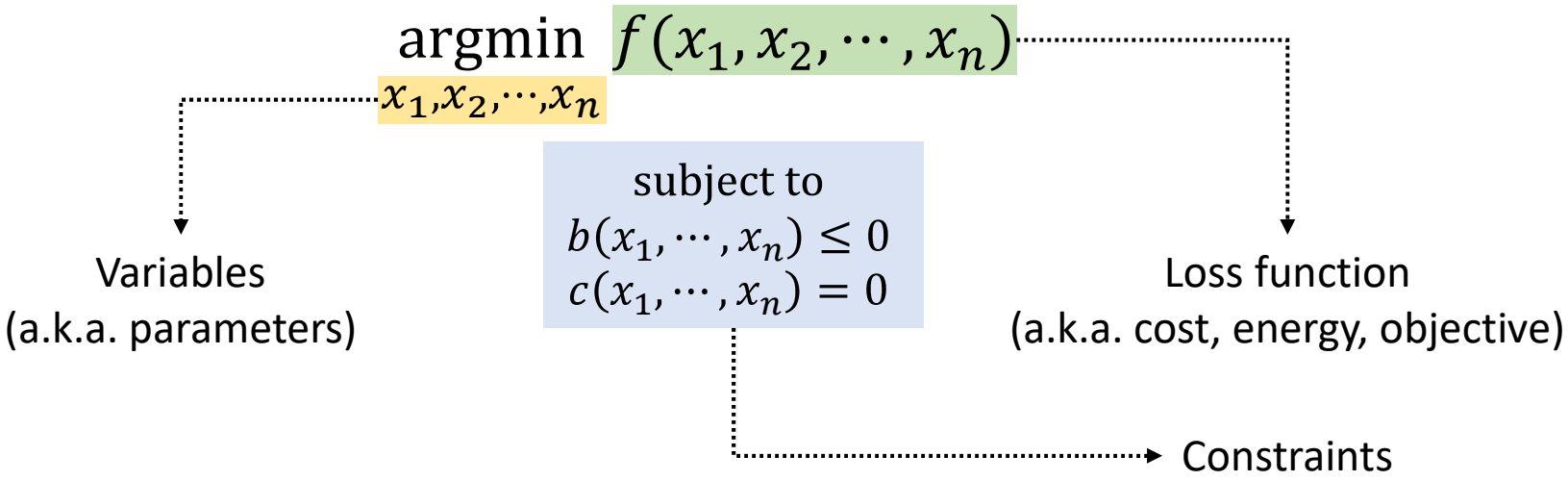
Variables
(a.k.a. parameters)

Loss function
(a.k.a. cost, energy, objective)



Find **a set of variables** $(x_1, x_2, \cdots, x_n)$ that give the minimum value of $f(x_1, x_2, \cdots, x_n)$

What is Optimization?



Find **a set of variables** $(x_1, x_2, \cdots, x_n)$ that give the minimum value of $f(x_1, x_2, \cdots, x_n)$ among the variables satisfying the constraints

What is Optimization?

- A minimization problem can be replaced by a maximization, and vice versa

$$\underset{x_1,x_2,\cdots,x_n}{\operatorname{argmin}} f(x_1,x_2,\cdots,x_n) \leftrightarrow \underset{x_1,x_2,\cdots,x_n}{\operatorname{argmax}} -f(x_1,x_2,\cdots,x_n)$$

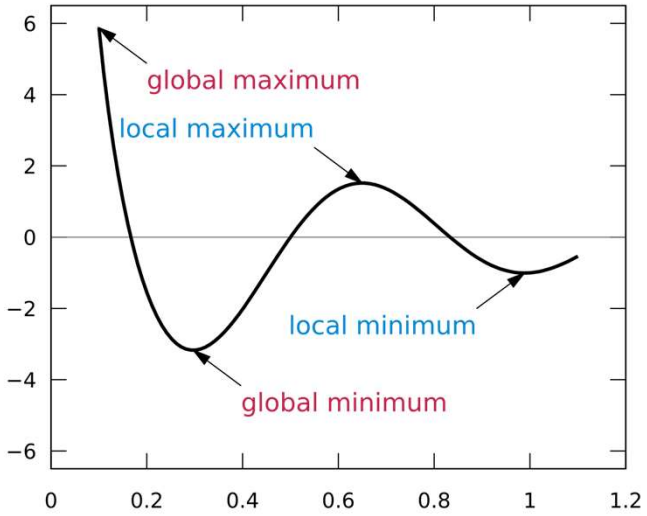
- Constants can be ignored while solving the problem

$$\underset{x_1,x_2,\cdots,x_n}{\operatorname{argmin}} f(x_1,x_2,\cdots,x_n) + c \leftrightarrow \underset{x_1,x_2,\cdots,x_n}{\operatorname{argmin}} f(x_1,x_2,\cdots,x_n)$$

Minima

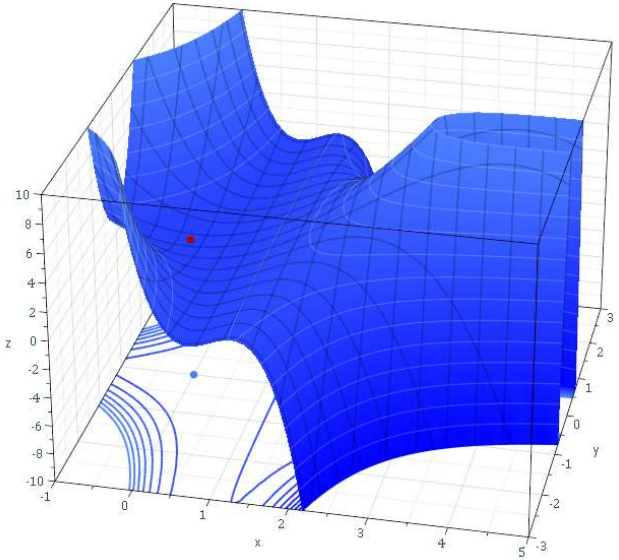
- A real value function f defined on X has a global minima x^* if $f(x^*) \leq f(x)$ for all $x \in X$

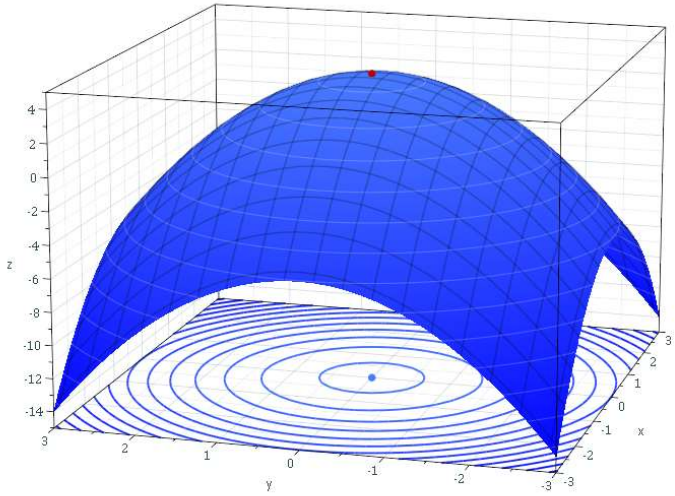
- A real value function f defined on X has a local minima if there exists ϵ such that $f(x^*) \leq f(x)$ for all x within distance ϵ of x^*



https://en.wikipedia.org/wiki/Maximum_and_minimum

Minima



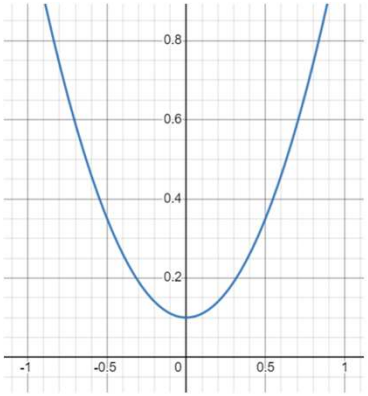


- In general, there could exists many minima/maxima given an objective function and constraints except for some special cases like the figure on the right

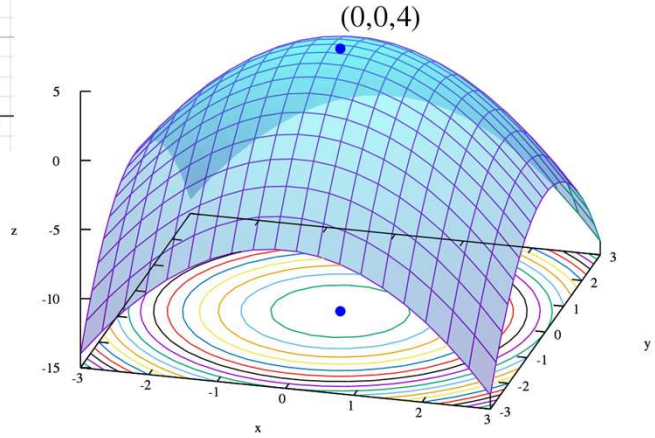
https://en.wikipedia.org/wiki/Maximum_and_minimum

Simple Examples

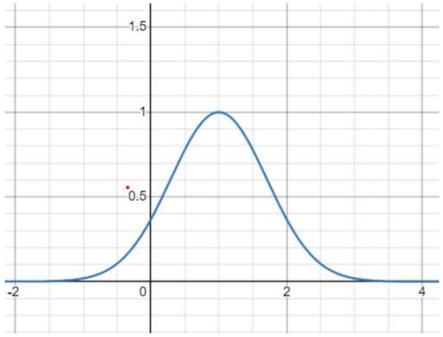
- $\min_{x}[x^2 + 0.1]$



- $\underset{x_1, x_2}{\text{argmax}}[-(x_1^2 + x_2^2) + 4]$



- $\underset{x}{\text{argmax}}[e^{-(x-1)^2}]$



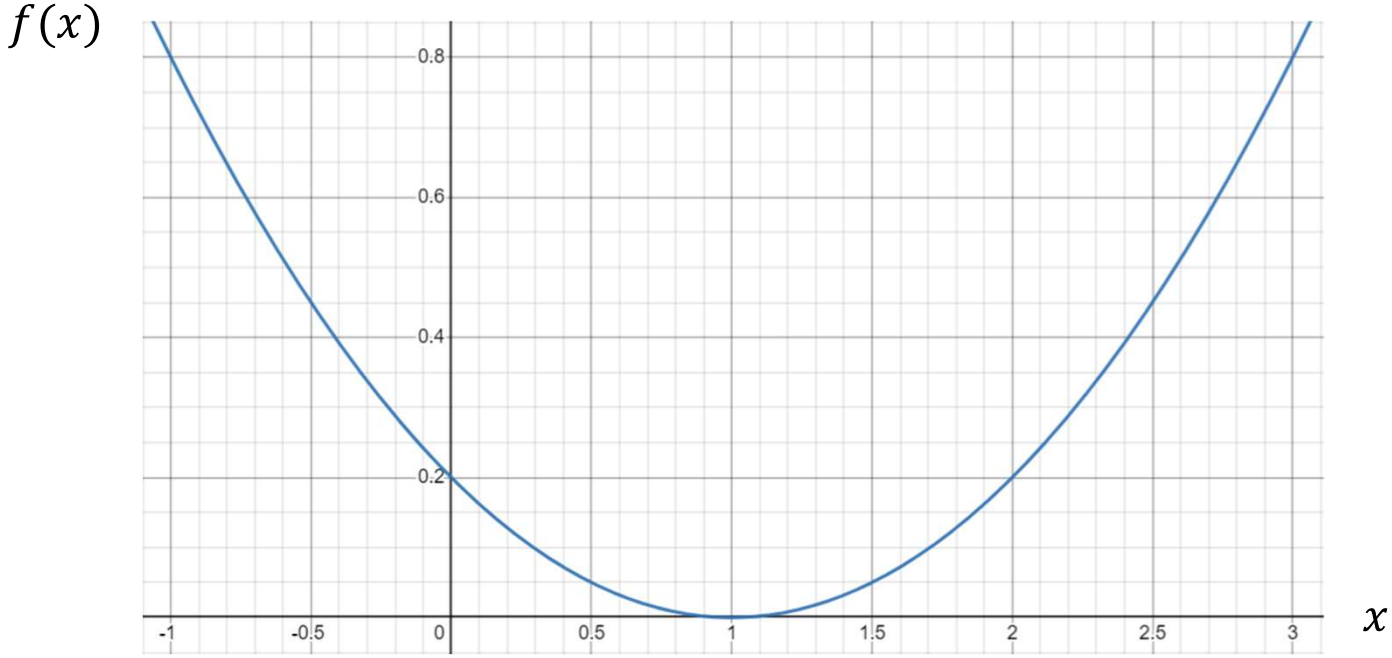
https://en.wikipedia.org/wiki/Mathematical_optimization#/

How to solve the problems?

- Analytical Methods
  - Unconstrained case
  - Lagrange Multipliers
  - KKT conditions
- Numerical Methods
  - Grid Search (brute-force)
  - Gradient-based Methods
  - Gradient-free Methods

Analytical Methods: An Example

$$\underset{x}{\operatorname{argmin}}\, f(x) = \underset{x}{\operatorname{argmin}} (x-1)^2$$



Analytical Methods: An Example

$$\underset{x}{\operatorname{argmin}} f(x) = \underset{x}{\operatorname{argmin}} (x-1)^2$$

$$\frac{df(x)}{dx} = \frac{d(x-1)^2}{dx} = 2(x-1) = 0$$

$$\therefore x = 1$$

- Geometrical meaning of critical points of which gradients are zero
  - Landscape of surrounding area looks flat
- But, this is a necessary condition to become a minimum and maximum

Gradient (First-order Derivative)

- The gradient of a scalar-valued differentiable function f of multiple variables is the vector-valued function $\nabla f(\mathbf{x})$ whose value at a point $\mathbf{x}$ is the "direction and rate of fastest increase“. - Wiki -

$$\nabla f(\mathbf{x}) = \nabla_{\mathbf{x}} f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}$$

Hessian (Second-order Derivative)

- The Hessian is a square matrix of second-order partial derivatives of a scalar-valued function. It describes the local curvature of a function of many variables. - Wiki -

$$\nabla^2 f(\mathbf{x}) = \nabla^2_{\mathbf{xx}} f = H_f(\mathbf{x}) = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_1 \partial x_n} \\ \dfrac{\partial^2 f}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} & \cdots & \dfrac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial^2 f}{\partial x_n \partial x_1} & \dfrac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

Analytical Methods: An Example

$$\underset{x_1,x_2}{\operatorname{argmin}}[(x_1-1)^2+3(x_2-2)^2]$$

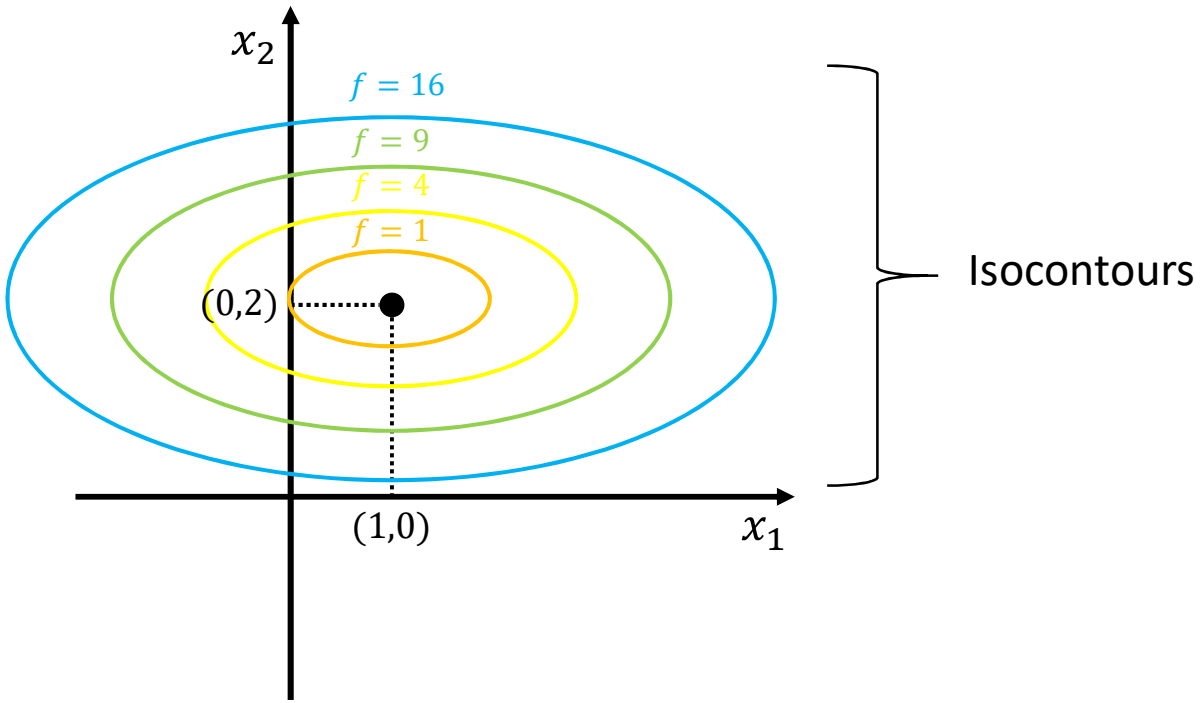
$$=\underset{x_1,x_2}{\operatorname{argmin}}\left[\begin{bmatrix}x_1 & x_2\end{bmatrix}\begin{bmatrix}1 & 0\\ 0 & 3\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}+\begin{bmatrix}-2 & -12\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix}+13\right]$$

$$\nabla f(x)=\begin{bmatrix}\dfrac{\partial f(x)}{\partial x_1}\\ \dfrac{\partial f(x)}{\partial x_2}\end{bmatrix}=\begin{bmatrix}2(x_1-1)\\ 6(x_2-2)\end{bmatrix}=\mathbf{0}$$

$$\therefore x_1=1, x_2=2$$

Isocontours

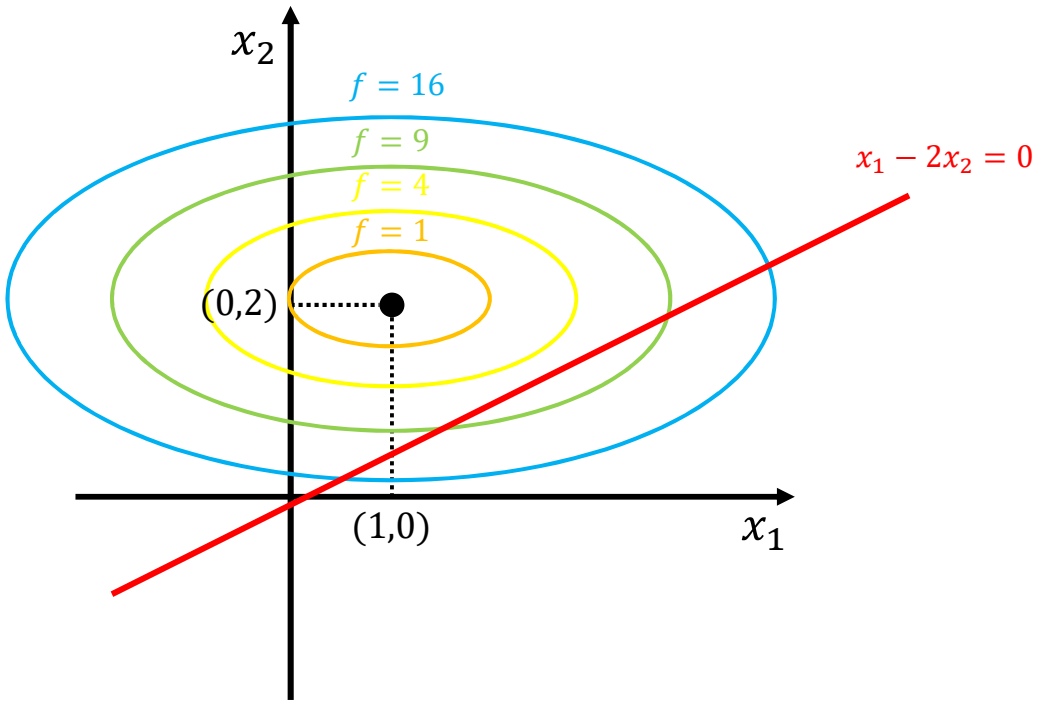
$$\underset{x_1,x_2}{\operatorname{argmin}}[(x_1-1)^2+3(x_2-2)^2]$$



Equality Constraints

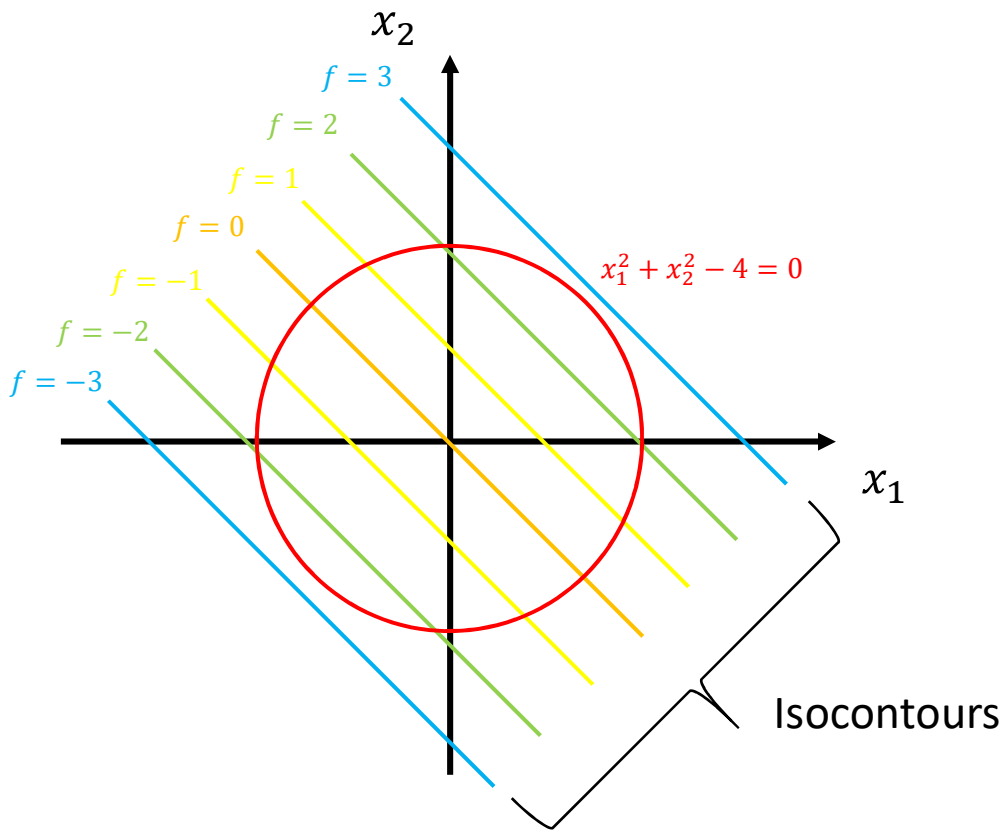
$$\underset{x_1,x_2}{\operatorname{argmin}}[(x_1-1)^2+3(x_2-2)^2]$$

$$\text{s.t. } x_1-2x_2=0$$

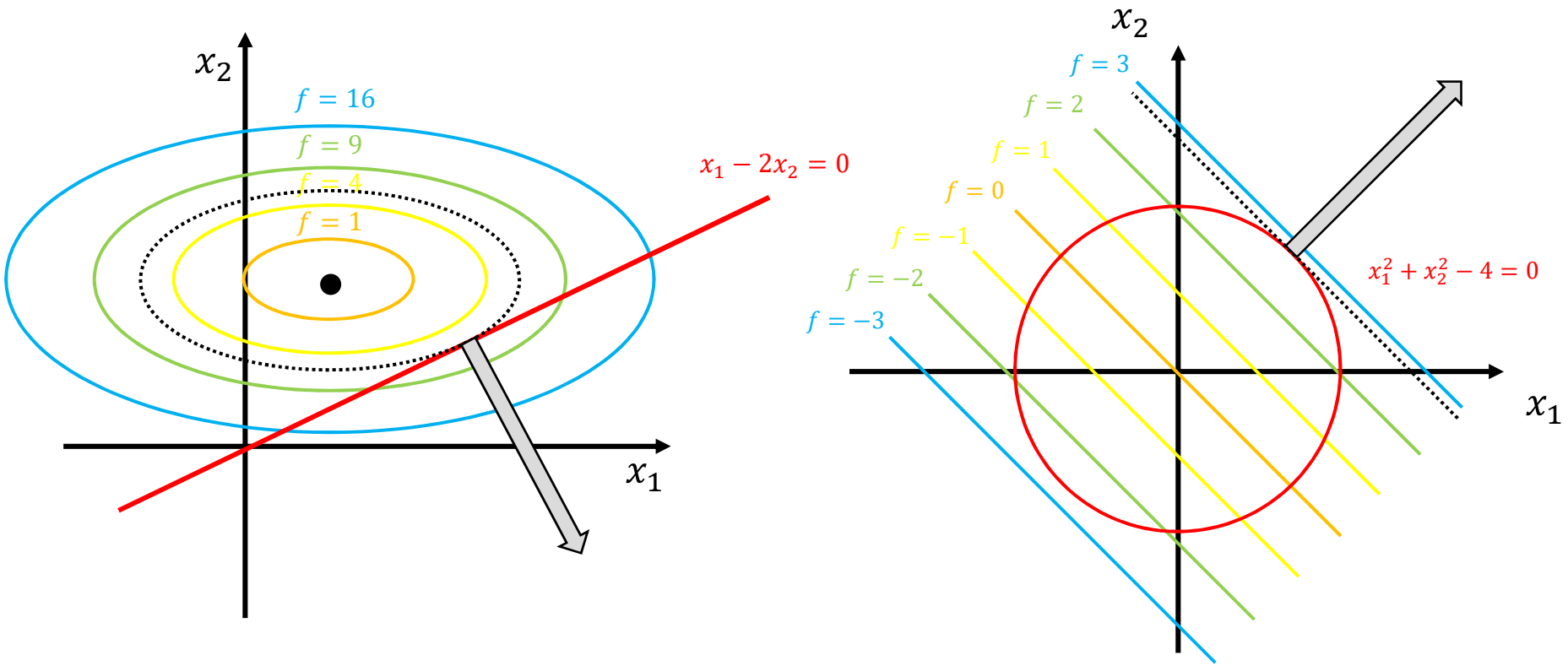


Equality Constraints

$$\underset{x_1,x_2}{\operatorname{argmax}}[x_1 + x_2]$$
$$\text{s.t. } x_1^2 + x_2^2 - 4 = 0$$



Equality Constraints



- At the minimum/maximum, the gradient of the objective function and the gradient of equality constraint are parallel

Lagrange Multipliers

- There should exist λ (Lagrange multiplier) which makes $\nabla f(\mathbf{x})$ and $\nabla c(\mathbf{x})$ the same

$$\nabla f(\mathbf{x}) - \lambda \nabla c(\mathbf{x}) = 0$$

- Similar statement holds true for the multiple constraints, refer to the doc below for the proof
  - https://personal.math.ubc.ca/~feldman/m226/multiLagrange.pdf

$$\nabla f(\mathbf{x}) - \sum \lambda_i \nabla c_i(\mathbf{x}) = 0$$

Lagrange Multipliers

- To find the solution the optimization problem with equality constraints

$$\underset{\mathbf{x}}{\operatorname{argmax}} f(\mathbf{x})$$
$$\text{subject to } c(\mathbf{x}) = 0$$

- Form a Lagrange function $L(\mathbf{x}, \lambda)$ then set its all partial derivatives (also including λ) as zero

$$L(\mathbf{x}, \lambda) = f(\mathbf{x}) - \lambda c(\mathbf{x})$$

$$\nabla_{\mathbf{x}} L(\mathbf{x}, \lambda) = 0 \rightarrow \nabla f(\mathbf{x}) = \lambda \nabla c(\mathbf{x})$$

$$\nabla_{\lambda} L(\mathbf{x}, \lambda) = 0 \rightarrow c(\mathbf{x}) = 0$$

Lagrange Multipliers

$$\underset{x_1,x_2}{\operatorname{argmax}}[x_1 + x_2] \text{ s.t. } x_1^2 + x_2^2 - 4 = 0$$

$$L(x_1, x_2, \lambda) = x_1 + x_2 - \lambda(x_1^2 + x_2^2 - 4)$$

$$\left.\begin{aligned} \nabla_{x_1} &= 1 - 2\lambda x_1 = 0 \\ \nabla_{x_2} &= 1 - 2\lambda x_2 = 0 \end{aligned}\right\} \; x_1 = x_2$$

$$\nabla_\lambda = x_1^2 + x_1^2 - 4 = 0$$

$$\therefore x_1 = x_2 = \sqrt{2}$$

Lagrange Multipliers

$$\underset{x_1,x_2}{\operatorname{argmax}}[(x_1-1)^2+3(x_2-2)^2]$$
$$\text{s.t. } x_1-2x_2=0$$

$$\therefore x_1=\frac{16}{7}, x_2=\frac{8}{7}$$

Lagrange Multipliers

$$\underset{x_1,x_2}{\operatorname{argmax}}[(x_1-1)^2+3(x_2-2)^2]$$
$$\text{s.t. } x_1-2x_2=0$$

$$\mathrm{L}(x_1,x_2,\lambda)=(x_1-1)^2+3(x_2-2)^2+\lambda(x_1-2x_2)$$

$$\left.\begin{aligned}\nabla_{x_1}&=2(x_1-1)+\lambda=0\\ \nabla_{x_2}&=6(x_2-2)-2\lambda=0\end{aligned}\right\}\ 2x_1+3x_2=8$$

$$\nabla_{\lambda}=x_1-2x_2=0$$

$$\therefore x_1=\frac{16}{7},x_2=\frac{8}{7}$$

Inequality Constraints

- The optimization problem with inequality constraints is harder to solve in general than the optimization with no constraint or equality constraints

- **Karush–Kuhn–Tucker (KKT)** conditions can be used to solve general problems but many problems with simple forms can be solved more efficiently

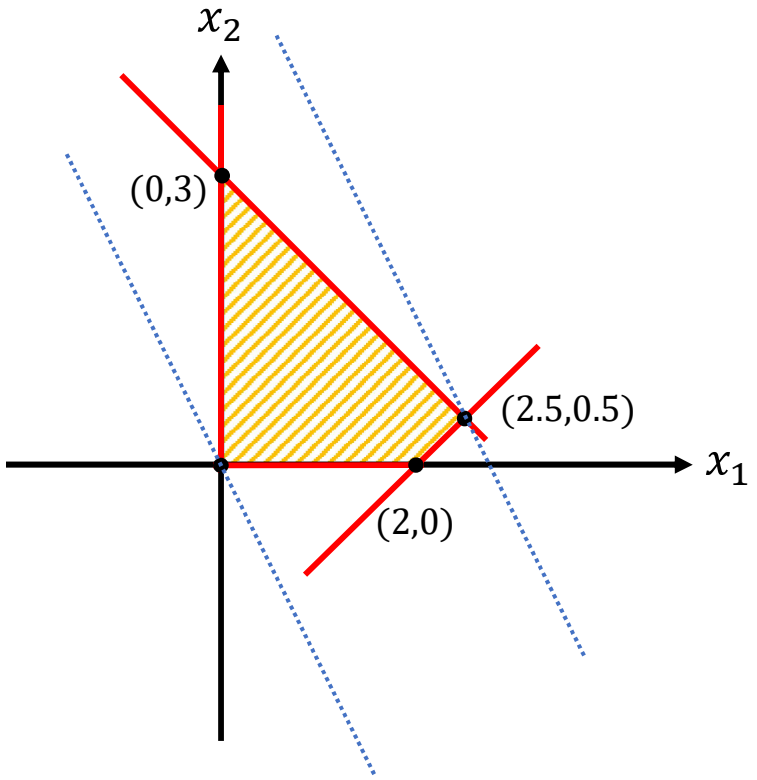
Linear Programming

- Linear Programing (LP) is a special case of the optimization problem where its objective function and constraints are linear

$$\begin{aligned} &\underset{\mathbf{x}}{\operatorname{argmin}}\ \mathbf{c}^{\mathrm{T}}\mathbf{x} \\ &\text{s.t. } A\mathbf{x} \leq \mathbf{b} \\ &\mathbf{x} \geq 0 \end{aligned}$$

Linear Programming: An Example

$$\underset{x_1,x_2}{\operatorname{argmax}}\ 2x_1 + x_2$$
$$\text{s.t. } x_1 + x_2 \leq 3$$
$$x_1 - x_2 \leq 2$$
$$x_1 \geq 0$$
$$x_2 \geq 0$$



- In LP, the solution is located at the ***boundaries***

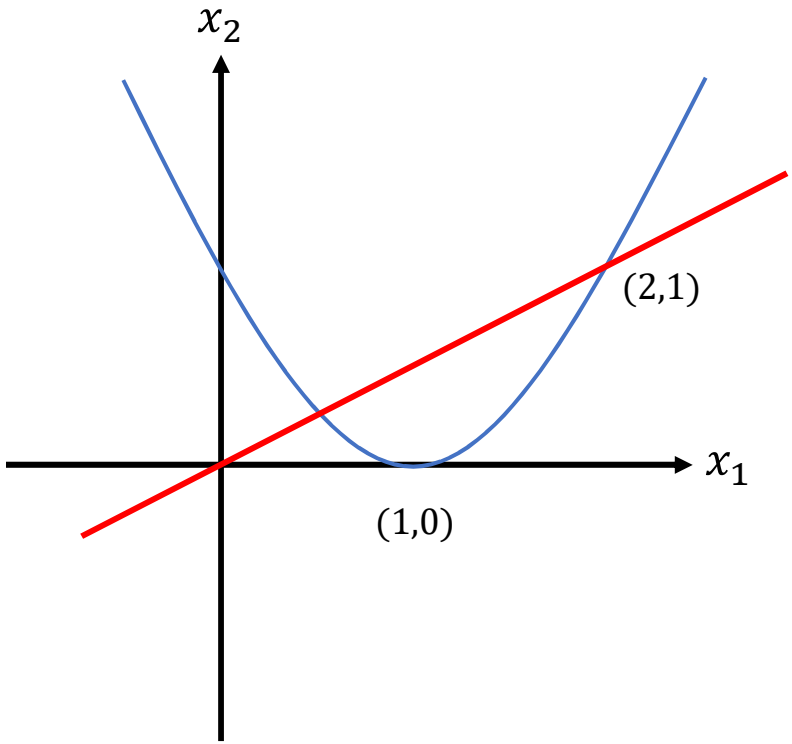
Quadratic Programming

- Quadratic Programing (QP) is a special case of the optimization problem where its objective function is quadratic and constraints are linear inequalities

$$\underset{\mathbf{x}}{\operatorname{argmin}} \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathrm{Q}\mathbf{x} + \mathbf{c}^{\mathrm{T}}\mathbf{x}$$
$$\text{s.t. } A\mathbf{x} \leq \mathbf{b}$$
$$\mathbf{x} \geq 0$$

Quadratic Programming: An Example

$$\underset{x}{\text{argmax}}(x-1)^2$$
$$\text{s.t. } x_1 - 2x_2 \leq 0$$



Quadratic Programming: An Example

- QP with linear equality constraints can be solved efficiently

$$\underset{\mathbf{x}}{\operatorname{argmin}} \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathrm{Q}\mathbf{x} - \mathbf{c}^{\mathrm{T}}\mathbf{x}$$
$$\text{s.t. } \mathrm{A}\mathbf{x} - \mathbf{b} = \mathbf{0}$$

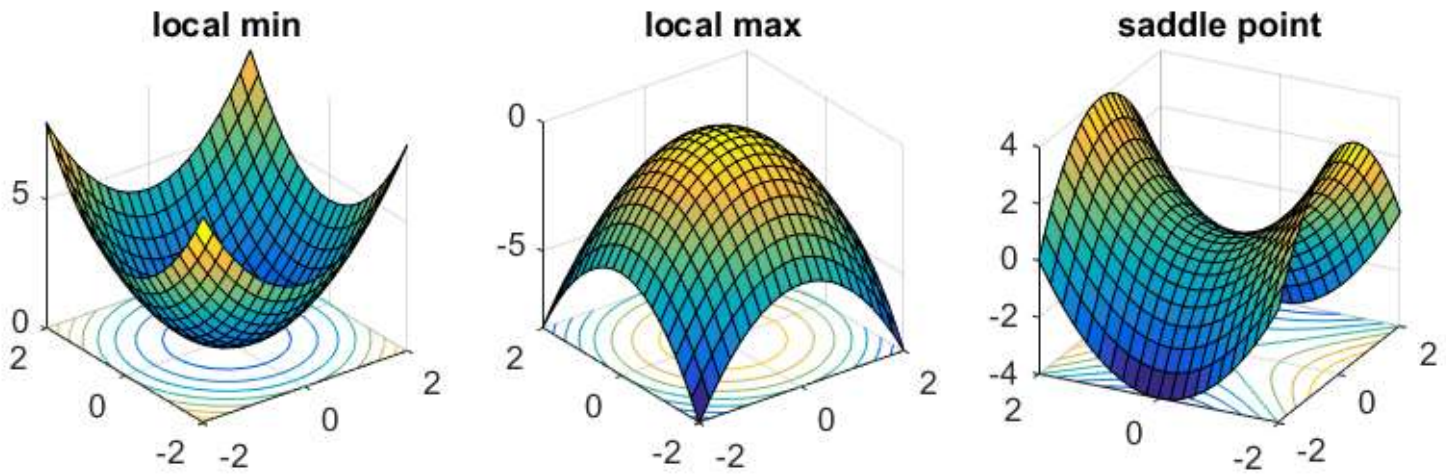
$$\begin{bmatrix} \mathrm{Q} & \mathrm{A}^{\mathrm{T}} \\ \mathrm{A} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{c} \\ \mathbf{b} \end{bmatrix}$$

: KKT condition

$$\therefore \begin{bmatrix} \mathbf{x} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathrm{Q} & \mathrm{A}^{\mathrm{T}} \\ \mathrm{A} & \mathbf{0} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{c} \\ \mathbf{b} \end{bmatrix}$$

Convexity

- To understand the type of a critical point $\mathbf{x}$ ($\nabla f(\mathbf{x}) = 0$), the second derivative (Hessian) can be used
  - Positive: Minima / Negative: Maxima / Zero: Saddle Pt.



https://www.offconvex.org/2016/03/22/saddlepoints/

Convexity

- Given a critical point $\mathbf{x}$ satisfies given constraints, if the function is **convex**, then it is guaranteed to be **the global minimum**

- A real-valued function is called **convex** if the line segment between any two points on the graph of the function lies **above** the graph between the two points.

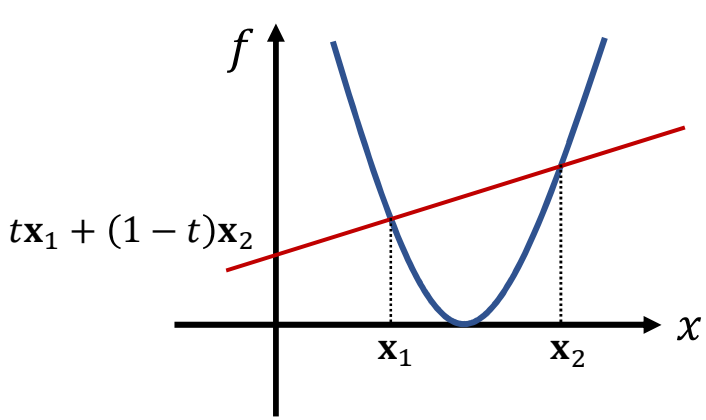
https://en.wikipedia.org/wiki/Maximum_and_minimum#/media/File:Extrema_example_original.svg

Convexity

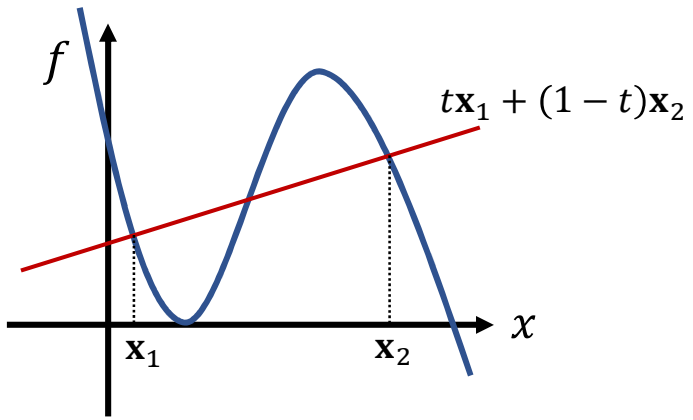
- Given a convex set X and a real valued function $\mathrm{f}: X \to R$, the function f is **convex** if and only if

$$f(t\mathbf{x}_1 + (1-t)\mathbf{x}_2) \leq t\mathrm{f}(\mathbf{x}_1) + (1-t)\mathrm{f}(\mathbf{x}_2)$$

$$\text{for all } 0 \leq \mathrm{t} \leq 1 \text{ and } \mathbf{x}_1, \mathbf{x}_2 \in X$$



Convex



Not convex

https://en.wikipedia.org/wiki/Maximum_and_minimum#/media/File:Extrema_example_original.svg

Numerical Methods

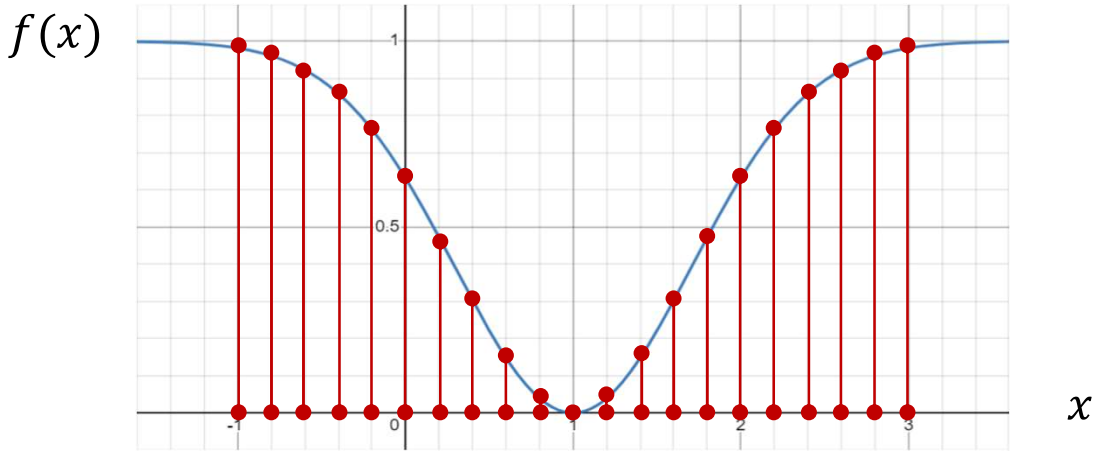
- Up to now, we learned how to solve the optimization problem analytically
- However, many practical problems can't be easily formulated as a combination of simple equations which can be solved by hand
  - E.g., 1000 optimization variables
- Instead, we develop an algorithm (procedure) to find one of the local minima

Numerical Methods

- Grid Search (brute-force)
- Gradient-based Methods
- Gradient-free Methods

Grid Search (Brute-Force)

$$\min_x f(x) = \max_x [e^{-(x-1)^2}]$$



Grid Search (Brute-Force)

- Pros
  - Guaranteed to find the minimum (assuming fine enough grid)
  - Simple and easy to implement

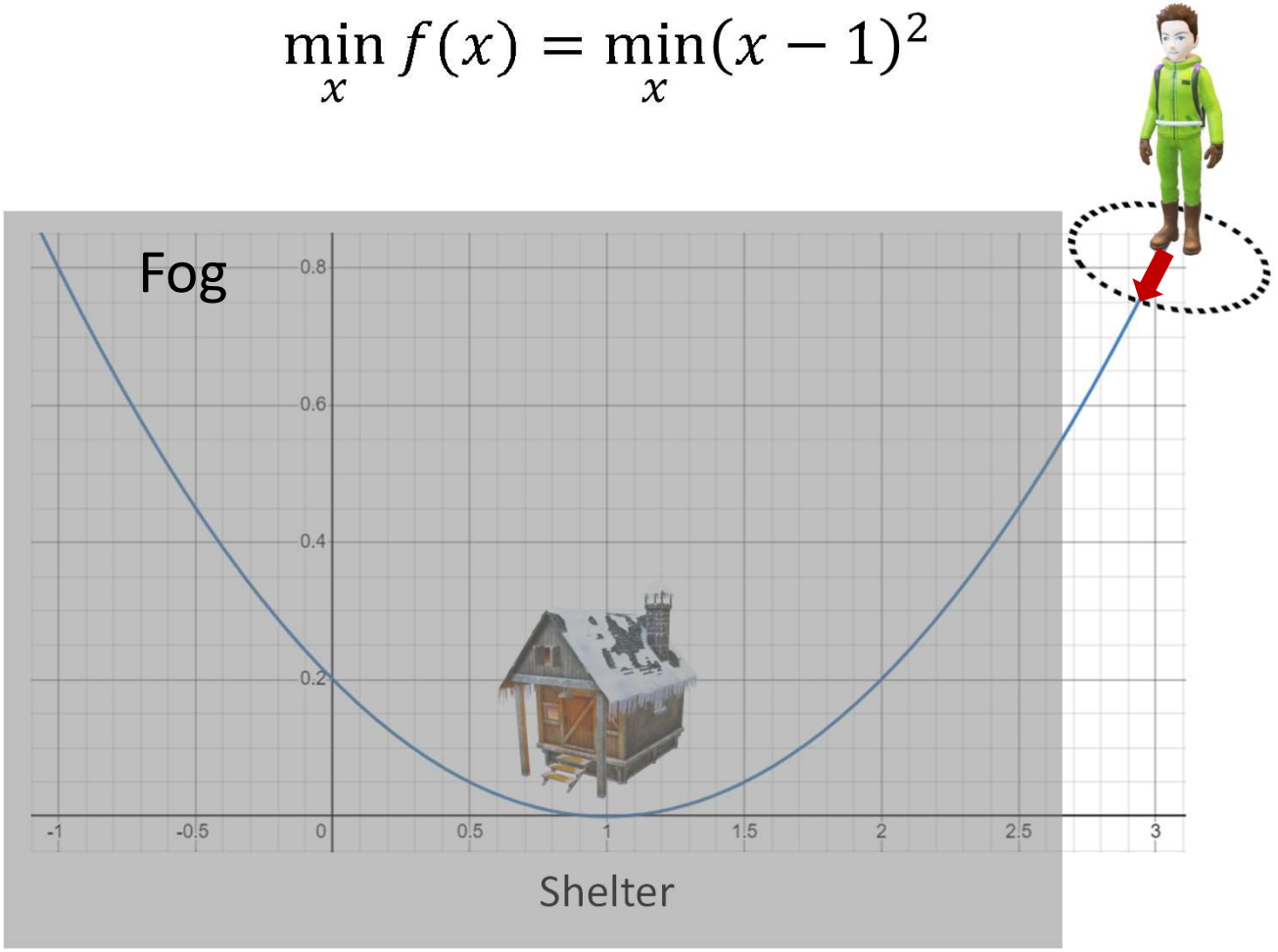
- Cons
  - Computationally inefficient
  - Only works for small number of parameters
    - 5 parameters and 100 slices => 10^10 evaluations are needed

Gradient-based Methods

- First-order vs. Second-order methods
  - The first order method only uses **gradient** (the first derivative) information only
  - The second order method uses **gradient** (the first derivative) and **Hessian** (the second derivative) informationType equation here.

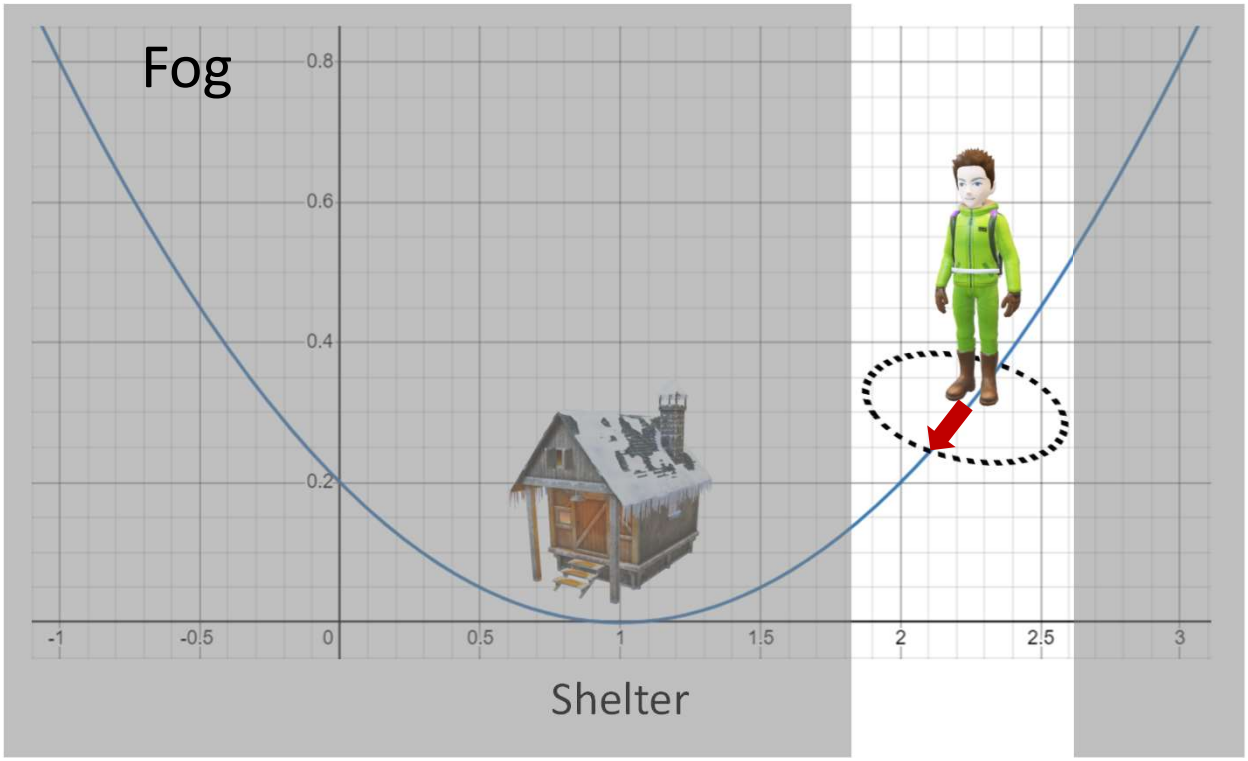
Gradient Descent

$$\min_x f(x) = \min_x (x-1)^2$$



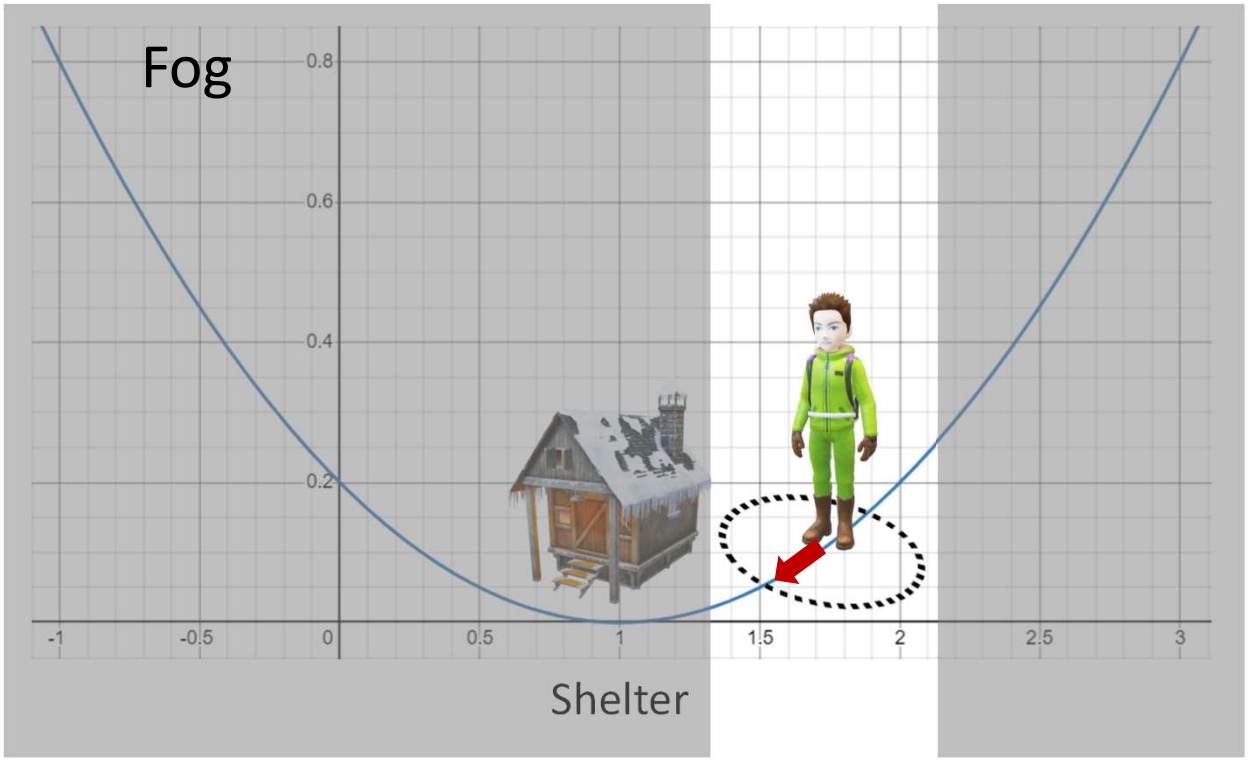
Gradient Descent

$$\min_x f(x) = \min_x (x-1)^2$$



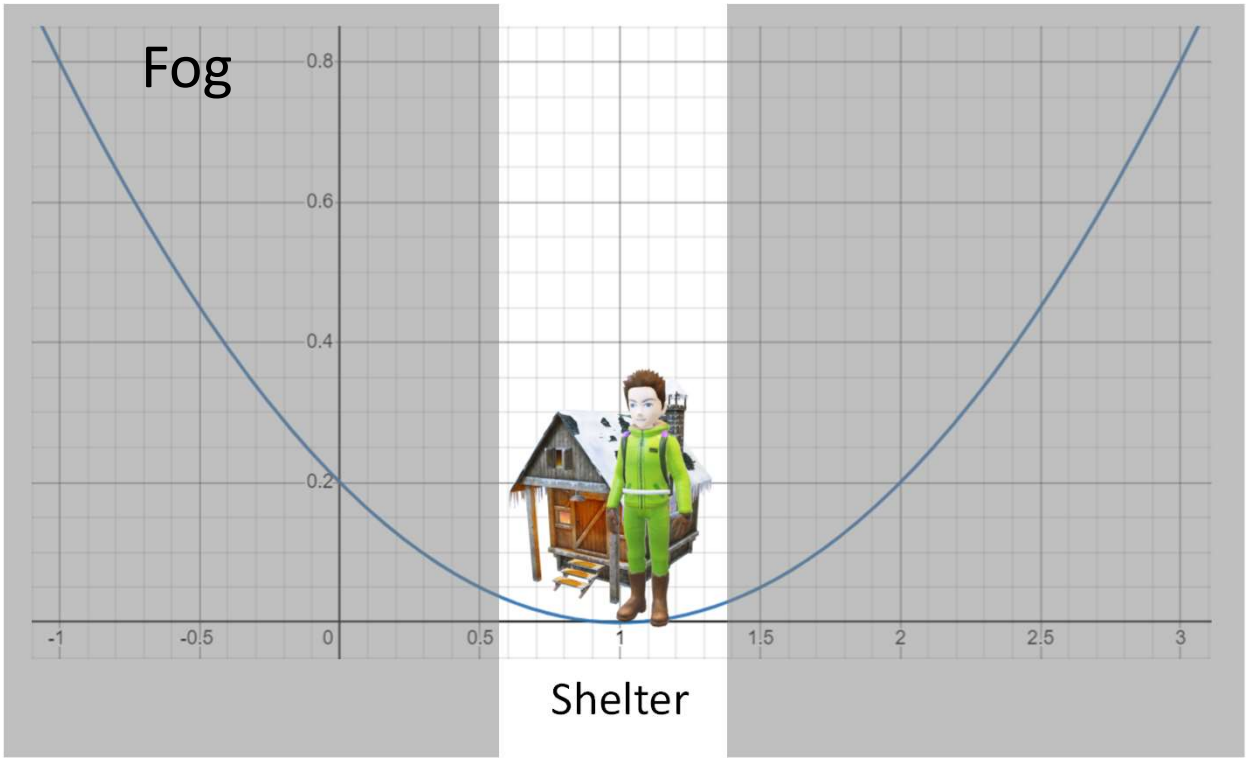
Gradient Descent

$$\min_x f(x) = \min_x (x-1)^2$$



Gradient Descent

$$\min_x f(x) = \min_x (x-1)^2$$



Gradient Descent

- It is a first-order iterative optimization algorithm for finding a local minimum of a differentiable function. The idea is to take repeated steps in the opposite direction of the gradient (or approximate gradient) of the function at the current point - Wikipedia -

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \gamma \nabla f(\mathbf{x}_t)$$

Step-size: Bravery

Gradient: Local Landscape at $\mathbf{x}_t$

Gradient Descent

- What if it is hard to calculate the gradient directly?

- We could use approximation
  - But, it requires extra number of evaluations on the objective function

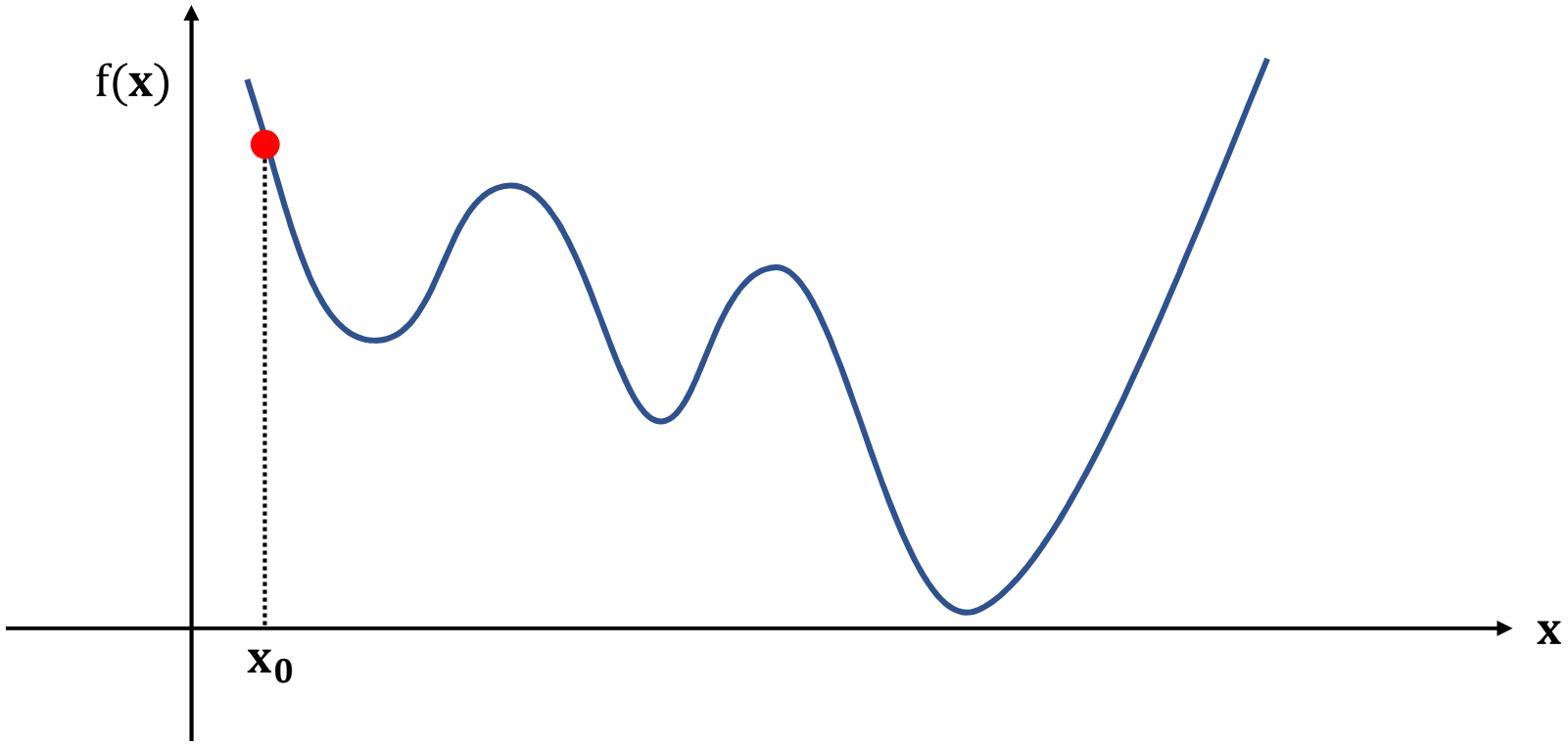
$$\nabla f(\mathbf{x}_t) \approx \frac{\nabla f(\mathbf{x}_t + \epsilon) - \nabla f(\mathbf{x}_t - \epsilon)}{2\epsilon}$$

Problems with Gradient Descent

- If the step-size is too small, gradient descent can be very slow
- If the step-size to too large, it can easily overshoot, fail to converge, or even diverge
- Choosing an appropriate step-size is important but not trivial
  - There exist many algorithms for how to change the step-size adaptively

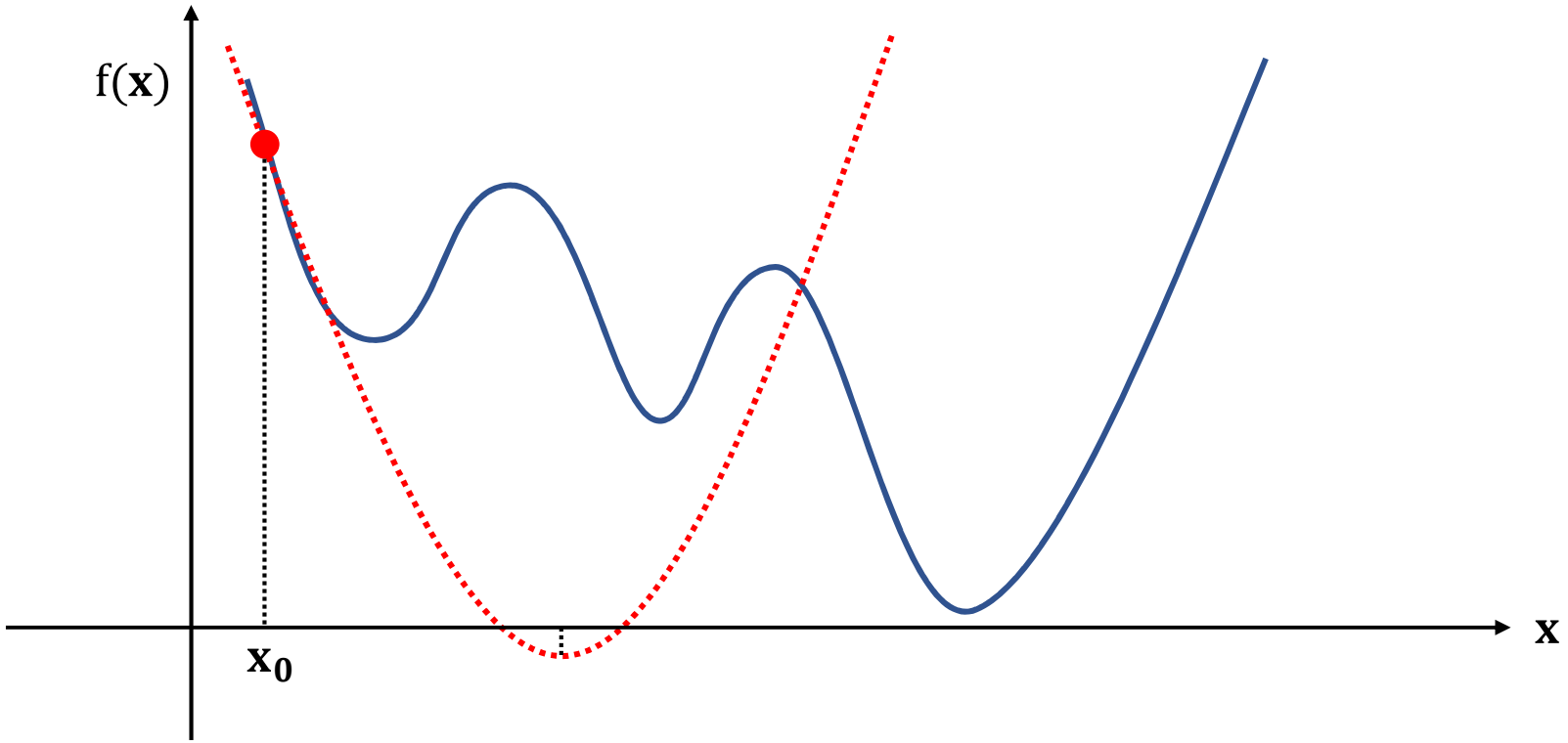
Sequential Quadratic Programming

- Sequential quadratic programming (SQP) finds a solution by iteratively solving a quadratic programming via linearizing the objective and constraint functions at each timestep



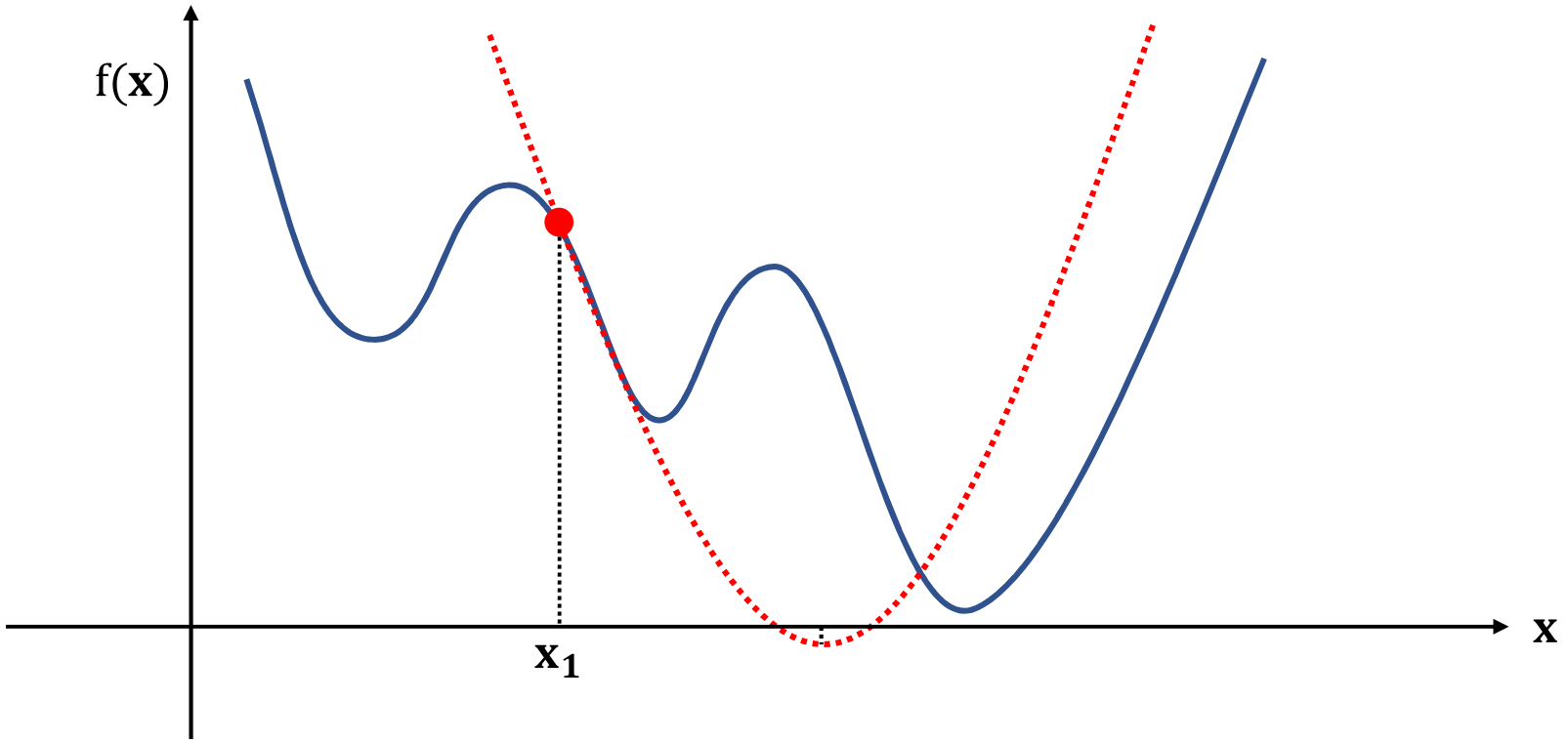
Sequential Quadratic Programming

- Sequential quadratic programming (SQP) finds a solution by iteratively solving a quadratic programming via linearizing the objective and constraint functions at each timestep



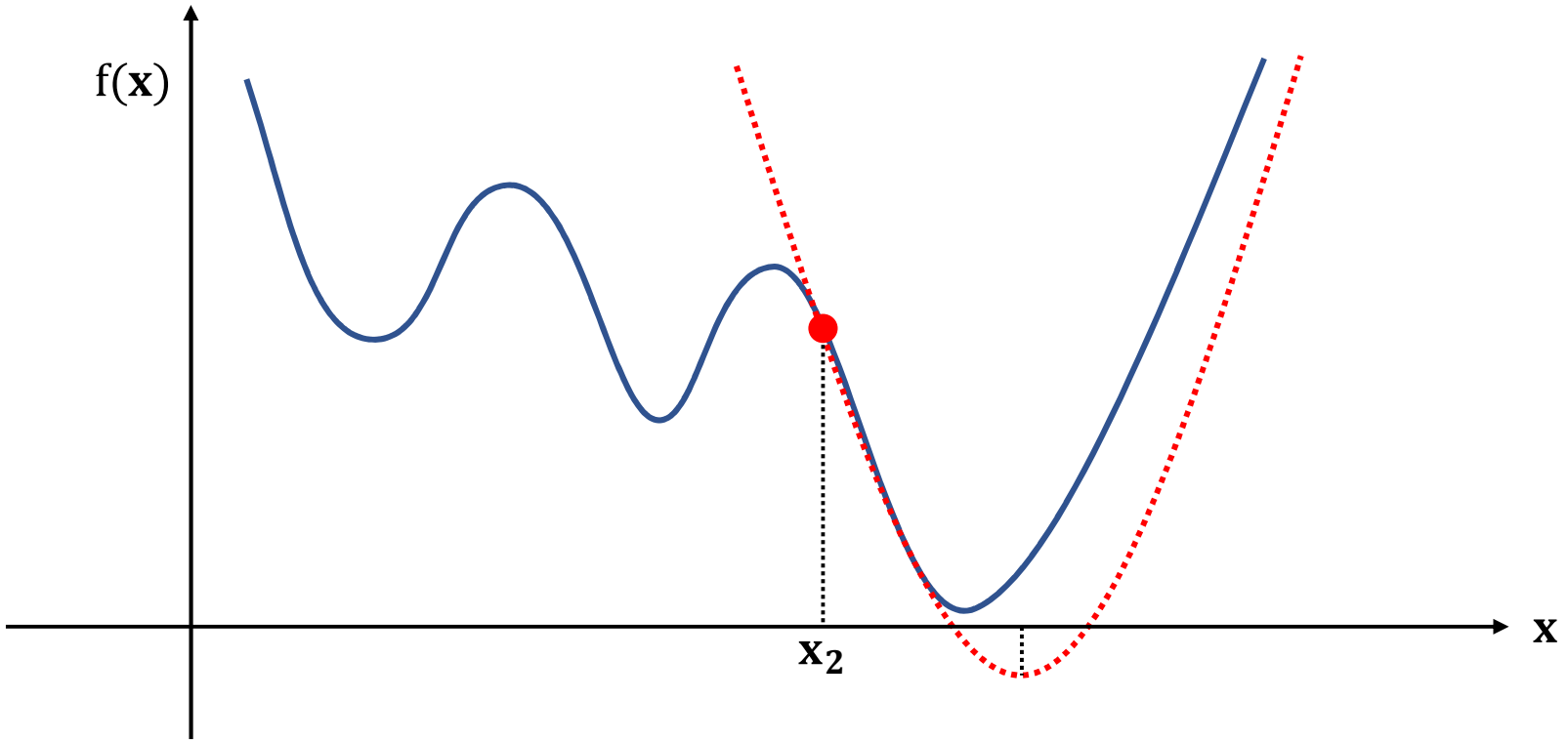
Sequential Quadratic Programming

- Sequential quadratic programming (SQP) finds a solution by iteratively solving a quadratic programming via linearizing the objective and constraint functions at each timestep



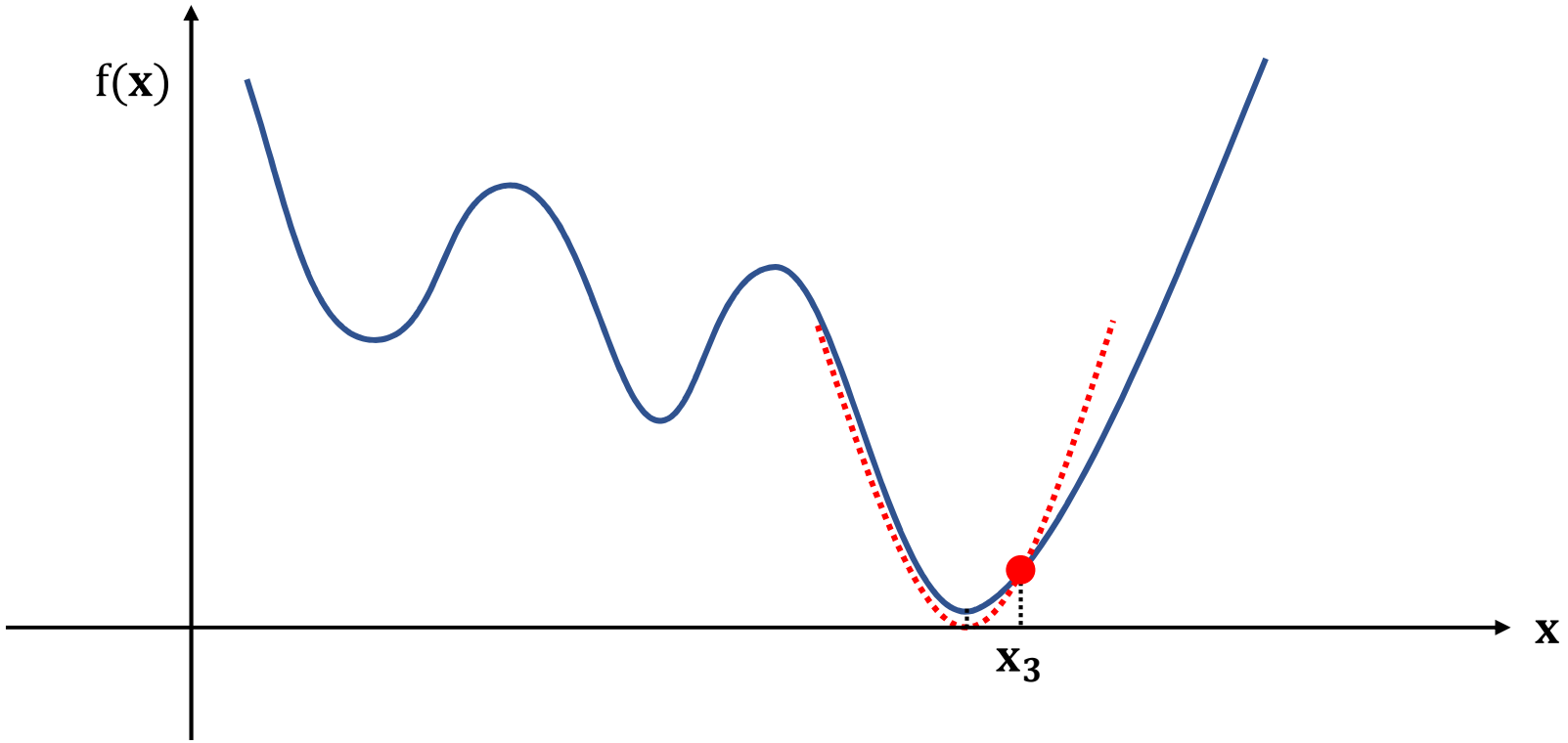
Sequential Quadratic Programming

- Sequential quadratic programming (SQP) finds a solution by iteratively solving a quadratic programming via linearizing the objective and constraint functions at each timestep



Sequential Quadratic Programming

- Sequential quadratic programming (SQP) finds a solution by iteratively solving a quadratic programming via linearizing the objective and constraint functions at each timestep



Sequential Quadratic Programming

$$\begin{aligned} &\underset{\mathbf{x}}{\operatorname{argmin}} f(\mathbf{x}) \\ &\text{subject to } b(\mathbf{x}_t) \leq \mathbf{0} \\ &c(\mathbf{x}_t) = \mathbf{0} \end{aligned}$$

Initialize $\mathbf{x}_t$

Until $|\mathbf{x}_{t+1} - \mathbf{x}_t| < \epsilon$

$$\mathbf{d} = \underset{\mathbf{d}}{\operatorname{argmin}} [f(\mathbf{x}_t) + \nabla f(\mathbf{x}_t)^{\mathrm{T}} \mathbf{d} + \frac{1}{2} \mathbf{d}^{\mathrm{T}} \nabla^2 f(\mathbf{x}_t) \mathbf{d}]$$

$$\text{s.t. } b(\mathbf{x}_t) + \nabla b(\mathbf{x}_t)^T \mathbf{d} \leq \mathbf{0}$$

$$c(\mathbf{x}_t) + \nabla c(\mathbf{x}_t)^T \mathbf{d} = \mathbf{0}$$

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \gamma \mathbf{d}$$

Constraints

- Projection Method
  - Solve QP with equality constraints only
  - If some variables do not satisfy the inequality constraints, perform projection

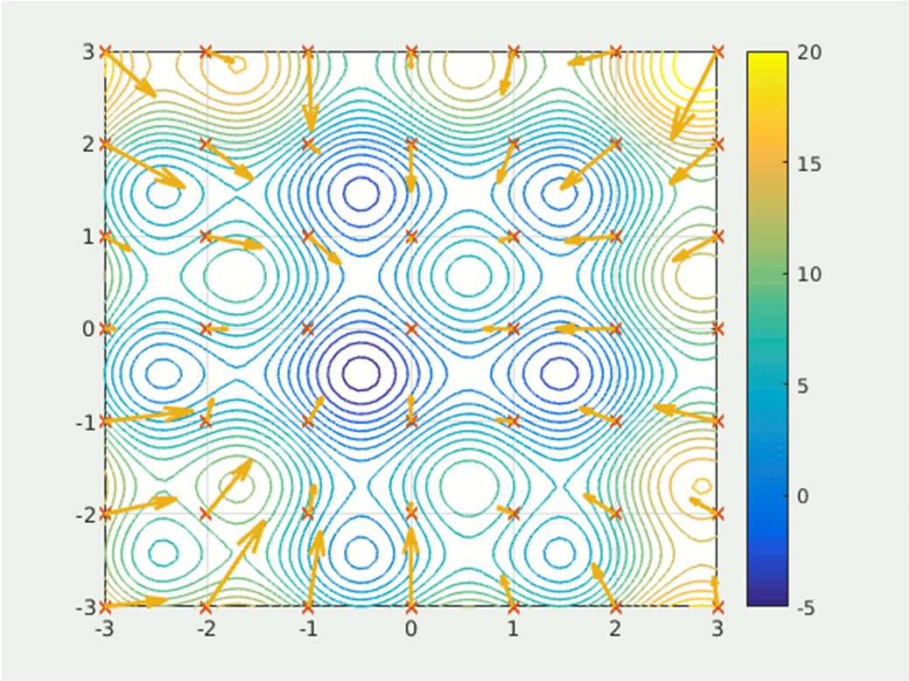
- Transfer from hard-constraints to soft-constraints
  - Reformulate the problem so that the constraints are included in the objective function

Gradient-free (Blackbox) Methods

- Particle Swarm
- Simulated Annealing
- Covariance Matrix Adaptation Evolution Strategy
- Genetic Algorithm

Particle Swarm Optimization

- PSO creates a set of solutions (**particles**) with velocities and iteratively moves these particles around in the search-space some criteria



Simulated Annealing

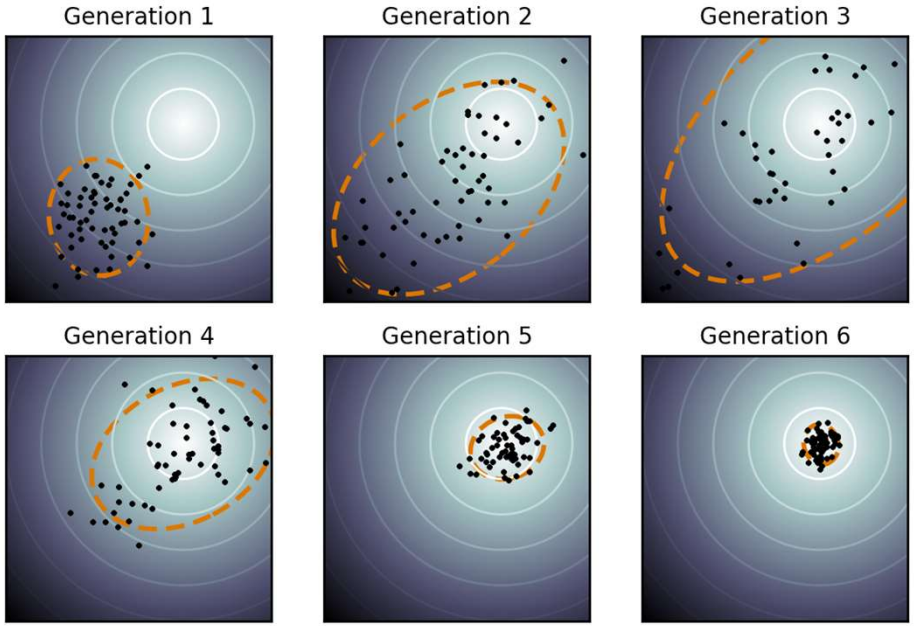
- An optimization algorithm that imitates the annealing process in metallurgy

Algorithm 1: Simulated Annealing (for maximization)

Initialize a solution x;
for $t = T_{max}$ *to* T_{min} **do**
  $f(x) \leftarrow$ Evaluate the current solution;
  $x' \leftarrow$ Select a new solution;
  $f(x') \leftarrow$ Evaluate the new solution;
  $\Delta x = f(x'_i) - f(x_i)$;
  $\rho = \frac{1}{1+exp(-\Delta x/t)}$;
  $x \leftarrow$ Accept a new solution with the probability ρ
end

CMA-ES

- Covariance Matrix Adaptation Evolution Strategy (CMA-ES) updates a stochastic distribution based on evolution strategy



Genetic Algorithm

- A method inspired by the process of natural selection process, where biologically inspired operators such as mutation, crossover, and selection are used to refine candidate solutions

Algorithm 1: GeneticAlgorithm

Initialize random population;
for $g = 1$ *to* g_{max} **do**
    Select parents from population;
    Perform **reproduction** (of children) from the selected parents;
    Perform **crossover** with some probability;
    Perform **mutation** with some probability;
    Reduce the population based on the evaluation ;
end

Summary

- We learned some fundamental concepts of the optimization problem, and some popular methods to solve the problem

- Nowadays, many efficient algorithms are readily available, so it is more important to understand which optimization solver is appropriate to your problems than you can implement specific algorithms unless you are a researcher who develop an algorithm